arXiv:1205.5003v1 [cs.MA] 22 May 2012

Ring Exploration with Oblivious Myopic Robots

Ajoy K. Datta[1], Anissa Lamani[2], Lawrence L. Larmore[1], and Franck Petit[3]
[1] School of Computer Science, University of Nevada Las Vegas, USA
[2] MIS, Université de Picardie Jules Verne Amiens, France
[3] LIP6, INRIA, CNRS, UPMC Sorbonne Universities, France

Abstract: The exploration problem in the discrete universe, using identical oblivious asynchronous robots without direct communication, has been well investigated. These robots have sensors that allow them to see their environment and move accordingly. However, the previous work on this problem assume that robots have an unlimited visibility, that is, they can see the position of all the other robots. In this paper, we consider deterministic exploration in an anonymous, unoriented ring using asynchronous, oblivious, and myopic robots. By myopic, we mean that the robots have only a limited visibility. We study the computational limits imposed by such robots and we show that under some conditions the exploration problem can still be solved. We study the cases where the robots visibility is limited to 1, 2, and 3 neighboring nodes, respectively.

Keywords: Asynchronous Anonymous Oblivious Robots, Deterministic Exploration, Discrete Environment, Limited Visibility

1 Introduction

There has been recent research on systems of autonomous mobile entities (hereafter referred to as *robots*) that must collaborate to accomplish a collective task. Possible applications for such multi-robot systems include environmental monitoring, large-scale construction, mapping, urban search and rescue, surface cleaning, risky area surrounding or surveillance, exploration of unknown environments, and other tasks in environments where instead of humans, robots are used. For many of these applications, the larger the number of robots is, the easier the implementation is.

However, the ability of a team of robots to succeed in accomplishing the assigned task greatly depends on the capabilities that the robots possess, namely, their sensing capabilities. Clearly, the type of viewing device has a great impact on the knowledge that the robots have of their environment. For example, if the robots have access to a global localization system (*e.g.*, GPS, egocentric zone-based RFID technology), then their viewing capabilities are *a priori* unlimited. By contrast, endowed with a camera or a sonar, vision capabilities are limited to a certain distance. Obviously, the stronger the device capabilities are, the easier the problem is solved.

In order to satisfy technological or budget constraints, it may be important to minimize both aforementioned parameters, *i.e.*, (*i*) the number of robots and (*ii*) capacities (or equipment) required to accomplish a given task.

In this paper, we consider both parameters for the *exploration* problem. Exploration is a basic building block for many applications. For instance, mapping of an unknown area requires that the robots (collectively) explore the whole area. Similarly, to search and rescue people after a disaster, the team of robots potentially has to explore the whole area. The so called "area" is often considered to be either the *continuous* Euclidean space (possibly with obstacles and objects) or a *discrete* space.

In the latter case, space is partitioned into a finite number of locations represented by a graph, where nodes represent indivisible locations that can be sensed by the robots, and where edges represent the possibility for a robot to move from one location to the other, *e.g.*, a building, a town, a factory, a mine, and more generally, zoned areas. In a discrete environment, exploration requires that starting from a configuration where no two robots occupy the same node, every node to be visited by at least one robot, with the additional constraint that all robots eventually stop moving.

We assume robots having weak capacities: they are *uniform* — meaning that all robots follow the same protocol —, *anonymous* — meaning that no robot can distinguish any two other robots, *oblivious* — they have no memory of any past behavior of themselves or any other robot, —, and *disoriented* — they have no labeling of direction. Furthermore, the robots have no (direct) means of communicating with each other, *i.e.*, they are unable to communicate together. However, robots are endowed with visibility sensors enabling to see robots located on nodes.

In this paper, we add another constraint: *myopia*. A myopic robot has limited visibility, *i.e.*, it cannot see the nodes located beyond a certain fixed distance ϕ. The stronger the myopia is, the smaller ϕ is. In other words, we consider that the strongest myopia corresponds to $\phi = 1$, when a robot can only see robots located at its own and at neighboring nodes. If $\phi = 2$, then a robot can see robots corresponding to $\phi = 1$ and the neighbors of its neighboring nodes. And so on. Note that the weaker myopia corresponds to $\phi = \lceil \frac{D}{2} \rceil - 1$, D being the diameter of the graph. Infinite visibility (*i.e.*, each robot is able to see the whole graph) is simply denoted by $\phi = \infty$.

We study the impact of myopia strength (*i.e.*, the size of the visibility radius) on the problem of exploration. This assumption is clearly motivated by limiting the vision capacities that each robot is required to have. As a matter of fact, beyond technological or budget constraints, it is more realistic to assume robots endowed with vision equipments able to sense their local (w.r.t. ϕ) environment than the global universe. Furthermore, solutions that work assuming the strongest assumptions also work assuming weaker assumptions. In our case, any distributed and deterministic algorithm that solves the exploration problem with robots having severe myopia ($\phi = 1$), also works with robots with lower myopia ($\phi > 1$), and even no myopia ($\phi = \infty$).

Related Work. Most of the literature on coordinated distributed robots assumes that the robots move in a *continuous* two-dimensional Euclidean space, and use visual sensors with perfect accuracy, permitting them to locate other robots with infinite precision, *e.g.*,[1, 2, 4, 7, 8, 17]. In each of these papers, other than [1, 7], the authors assume that each robot is able to see all other robots in the plane. In [1], the authors give an algorithm for myopic robots with limited visibility, converging toward a single point that is not known in advance. The convergence problem is similar to the gathering problem where the robots must meet in a single location in finite time. In [7], the authors present a gathering algorithm for myopic robots in the plane, which requires that the robots agree on a common coordinate system.

In the discrete model, *gathering* and *exploration* are the two main problems that have been investigated so far *e.g.*, [9, 10, 12, 13] for the gathering problem and [3, 5, 6, 14] for the exploration problem. In [12], the authors prove that the gathering problem is not feasible in some symmetric configurations and propose a protocol based on breaking the symmetry of the system. By contrast, in [13], the authors propose a gathering protocol that exploits this symmetry for a large number of robots ($k > 18$) closing the open problem of characterizing symmetric situations on the ring that admit gathering solutions. In [10, 11], the authors achieve similar results assuming weaker robot capabilities: robots may not be able to count the number of robots on the same node. In [9], the authors studied the gathering problem considering robots having only a local visibility *ie,*

they can only see robots located at its own and at adjacent nodes, *i.e.*, ϕ is assumed to be equal to 1. Under this assumption, the authors provide a complete solution of the gathering problem for regular bipartite graphs. They first characterize the class of initial configurations allowing to solve the gathering problem on such graphs, and they propose a gathering algorithm assuming that the system starts from a configuration in this class.

In [6], it is shown that, in general, $\Omega(n)$ robots are necessary to explore a tree network of n nodes. In [5], it is proved that no deterministic exploration is possible on a ring when the number of robots k divides the number of nodes n. In the same paper, the authors proposed a deterministic algorithm that solves the problem using at least 17 robots, provided that n and k are co-prime. In [3], it is shown that no protocol (probabilistic or deterministic) can explore a ring with fewer than four robots. In the same paper, the authors give a probabilistic algorithm that solves the problem on a ring of size $n > 8$ that is optimal for the number of robots. Finally, in [14], the authors reduce the gap in the deterministic case between a large upper bound ($k \geq 17$) and a small lower bound ($k > 3$) by showing that 5 robots are necessary and sufficient in the case that the size of the ring is even, and that 5 robots are sufficient when the size of the ring is odd.

Contribution. To the best of our knowledge, all previous results for discrete versions of the exploration problem assume unlimited visibility ($\phi = \infty$), *i.e.*, the whole graph is seen by each robot. In this paper, we consider deterministic algorithms for ring networks of n nodes that uses k myopic robots for ring networks. Following the same arguments as in [9], it should be emphasized that exploration would not be solvable starting from any configuration but from configurations where each robot is at distance at most ϕ of another one—at least when $\phi = 1$.

Our contribution is threefold. First, we tackle the case where robots have a visibility $\phi = 1$, *i.e.*, they can only see the state of nodes at distance 1. In this case, we show that the exploration problem is not solvable in both semi-synchronous and asynchronous models, for $n > 1$ and $1 \leq k \leq n$. Next, we show that even in the (fully) synchronous model, the exploration problem cannot be solved with less than 5 robots when $n > 6$. We then propose, optimal deterministic algorithms in the synchronous model for both cases $3 \leq n \leq 6$ and $n > 6$. Second, we propose two deterministic solutions in the asynchronous model when robots have a visibility $\phi = 2$ using respectively 7 (provided $n > k$) and 9 (provided that $n > k\phi + 1$) robots. Finally, we show that no exploration is possible with less than 5 robots when they have a visibility $\phi = 3$ and $n > 13$ in both semi-synchronous and asynchronous model. We then propose two asynchronous solutions that solve the problem using respectively 5 and 7 robots. The former works starting from specific configurations. All our solutions work assuming that each robot is at distance at most ϕ of another one. Both solutions for 7 robots with $\phi = 2$ and 5 robots with $\phi = 3$ work starting from specific configurations.

Roadmap. Section 2 presents the system model that we use throughout the paper. We present our results for the cases $\phi = 1$, $\phi = 2$, and $\phi = 3$ in Section 3, 4, and 5, respectively. Section 6 gives some concluding remarks.

2 Model

System. We consider systems of autonomous mobile entities called *robots* moving into a discrete environment modeled by a graph $G = (V, E)$, V being a set of nodes representing a set of *locations* (or *stations*) where robots are, E being a set of edges that represent bidirectional connections through

which robots move from a station to another. We assume that the graph G is a *ring* of n nodes, $u_0, \ldots, u_{n-1}$, *i.e.*, u_i is connected to both u_{i-1} and u_{i+1} — every computation over indices is assumed to be modulo n. The indices $0 \ldots i \ldots n-1$ are used for notation purposes only; the nodes are *anonymous* and the ring is *unoriented*, *i.e.*, given two neighboring nodes u, v, there is no kind of explicit or implicit labeling allowing to determine whether u is on the right or on the left of v. On this ring, $k \leq n$ anonymous robots $r_0, \ldots, r_j, \ldots, r_{k-1}$ are collaborating together to achieve a given task. The indices $0 \ldots j \ldots k-1$ are used for notation purposes only, since they are undistinguished. Additionally, the robots are *oblivious*, *i.e*, they have no memory of their past actions. We assume the robots do not communicate in an explicit way. However, they have the ability to sense their environment. Zero, one, or more robots can be located on a node. The number of robots located on a node u_i at instant t is called *multiplicity* of u_i and is denoted by $M_i(t)$ (or simply M_i, if t is understood). We say a node u_i is *free* at instant t if $M_i(t) = 0$. Conversely, we say that u_i is occupied at instant t if $M_i(t) \neq 0$. If $M_i(t) > 1$ then, we say that there is an $M_i(t).tower$ (or simply a *tower*) at u_i at instant t.

We assume that each robot is equipped with an abstract device called *multiplicity sensor* allowing to measure node multiplicity. Multiplicity sensors are assumed to have *limited capacities* (so called, *limited visibility*), *i.e.*, each robot r_j can sense the multiplicity of nodes that are at most at a fixed distance[1] ϕ ($\phi > 0$) from the node where r_j is located. We assume that ϕ is a common value to all the robots. The ring being unoriented, no robot is able to give an orientation to its view. More precisely, given a robot r_j located at a node u_i, the multiplicity sensor of r_j outputs a sequence, s_j, of $2\phi+1$ integers $x_{-\phi}, x_{1-\phi}, \ldots, x_{-1}, x_0, x_1, \ldots, x_{\phi-1}, x_\phi$ such that:

either $x_{-\phi} = M_{i-\phi}, \ldots, x_0 = M_i, \ldots, x_\phi = M_{i+\phi}$,
or $x_{-\phi} = M_{i+\phi}, \ldots, x_0 = M_i, \ldots, x_\phi = M_{i-\phi}$.

If the sequence $x_1, \ldots, x_{\phi-1}, x_\phi$ is equal to the sequence $x_{-1}, \ldots, x_{1-\phi}, x_{-\phi}$, then the view of r_j is said to be *symmetric*. Otherwise, it is said to be *asymmetric*.

Computations. Robots operate in three phase cycles: Look, Compute and Move (L-C-M). During the Look phase, a robot r_j located at u_i takes a snapshot of its environment given by the output of its multiplicity sensors, *i,e.*, a sequence s_j of $2\phi+1$ integers. Then, using s_j, r_j computes a destination to move to, *i.e.*, either u_{i-1}, u_i, or u_{i+1}. In the last phase (move phase), r_j moves to the target destination computed in the previous phase.

Given an arbitrary orientation of the ring and a node u_i, $\gamma^{+i}(t)$ (resp., $\gamma^{-i}(t)$) denotes the sequence $\langle M_i(t)M_{i+1}(t)\ldots M_{i+n-1}(t)\rangle$ (resp., $\langle M_i(t)\ldots M_{i-(n-1)}(t)\rangle$). We call the sequences $\gamma^{-i}(t)$ and $\gamma^{+i}(t)$ *mirrors* of each other. Of course, a symmetric sequence is its own mirror. By convention, we state that the *configuration* of the system at instant t is $\gamma^0(t)$. Let $\gamma = \langle M_0 M_1 \ldots M_{n-1}\rangle$ be a configuration. The configuration $\langle M_i, M_{i+1}, \ldots, M_{i+n-1}\rangle$ is obtained by rotating γ of $i \in [0 \ldots n-1]$. Two configurations γ and γ' are said to be *undistinguishable* if and only if γ' is obtained by rotating γ or its mirror. Two configurations that are not undistinguished are said to be *distinguished*.

We call a configuration $\gamma = \langle M_0 \ldots M_{n-1}\rangle$ *towerless* if $M_i \leq 1$. for all i. A configuration at which no robot that can move we call *terminal*.

An *inter-distance* d refers to the minimum distance taken among distances between each pair of distinct robots. Given a configuration γ, a *d.block* is any maximal elementary path in which robots are at distance d. Each occupied node at the extremity of the d.block is called a *border*. The *size* of a d.block is the number of robots in the d.block. A robot not being into a d.block is said to be *isolated*.

[1] The distance between two vertices in the ring is the number of edges in a shortest path connecting them.

A ϕ*.group* is any maximal elementary path in which there is one robot every node at distance at most ϕ of each other. In other words, a ϕ.group is a d-block for the particular case of $d = \phi$.

At each step t, a non-empty subset of robots is *selected* the *scheduler*, or *daemon*. The scheduler is viewed as an abstract external entity. We assume a *distributed fair* scheduler. Distributed means that, at every instant, any non-empty subset of robots can be activated. Fair means that every robot is activated infinitely often during a computation. We consider three computational models: (i) The *semi-synchronous* model, (ii) the (*fully*) *synchronous* model, and (iii) the *asynchronous* model. In the former model, at every time instant t, every robot that is selected instantaneously executes the full cycle L-C-M between t and $t + 1$. (This model is known as the ATOM model [17]). The synchronous model is similar to the semi-synchronous model, except that the scheduler selects all enabled robots at each step. In the asynchronous model, cycles L-C-M are performed asynchronously for each robot, *i.e.*, the time between Look, Compute, and Move operations is finite but unbounded, and is decided by the scheduler for each action of each robot. Note that since each robot is assumed to be located at a node, the model considered in our case can be seen as *CORDA* [15] with the following constraint: the Move operation is *atomic*, *i.e*, no robot can be located on an edge. In other words, whenever a robot takes a snapshot of its environment, the other robots are always seen on nodes, never on edges. Since the scheduler is allowed to interleave the operations, a robot can move according to an outdated view, *i.e.*, during the computation phase, some robots may have moved.

We call *computation* any infinite sequence of configurations $\gamma_0, \ldots, \gamma_t, \gamma_{t+1}, \ldots$ such that (1) γ_0 is a possible initial configuration and (2) for every instant $t \geq 0$, γ_{t+1} is obtained from γ_t after some non-empty set of robots executes an action. Any transition γ_t, γ_{t+1} is called a step of the computation. A computation c *terminates* if c contains a terminal configuration.

Algorithm. Each rule in the algorithm is presented in the following manner:
$< Label > < Guard > :: < Statement >$. The guard is a possible sequence s provided by the sensor of a robot r_j:$s = x_{i-\phi}, \ldots, x_{i-1}, (x_i), x_{i+1}, \ldots, x_{i+\phi}$. A robot r_j at node u_i is *enabled* at time t (or simply *enabled* when it is clear from the context) if:
$s = M_{i-\phi}(t), \ldots, M_{i-1}(t), (M_i), M_{i+1}(t), \ldots, M_{i+\phi}(t)$, or
$s = M_{i+\phi}(t), \ldots, M_{i+1}(t), (M_i), M_{i-1}(t), \ldots, M_{i-\phi}(t)$. The corresponding rule $< Label >$ is then also said to be *enabled*. The statement describes the action to be performed by r_j. There are only two possible actions: (i) $\rightarrow$, meaning that r_j moves towards the node u_{i+1}, (ii) $\leftarrow$, meaning that r_j moves towards the node u_{i-1}. Note that when the view of r_j is symmetric, the scheduler chooses the action to be performed. In this case, we write statement: $\leftarrow \vee \rightarrow$.

Character `'?'` in the algorithms means *any value*.

Problem Specification.

Definition 2.1 (Exploration) *Let $\mathcal{P}$ be a* deterministic *protocol designed for a team of k robots with a given positive visibility ϕ, evolving on an n-size ring. $\mathcal{P}$ is a* deterministic *(*terminating*)* exploration *protocol if and only if every computation c of $\mathcal{P}$ starting from any towerless configuration, the k robots collectively explore the ring, i.e., (i) c terminates in* finite time*, (ii) Every node is visited by at least one robot during c.*

Theorem 2.1 *If $\mathcal{P}$ is a deterministic exploration protocol, then for every pair of distinct configurations γ_i and γ_j in any execution c of $\mathcal{P}$, γ_i and γ_j are distinguished.*

Proof. Suppose $\mathcal{P}$ is a deterministic exploration protocol, and there exists an execution c of $\mathcal{P}$ and a pair of distinct configurations, γ_i and γ_j of c, such that γ_i and γ_j are undistinguished. Since $\mathcal{P}$ is a deterministic exploration protocol, c terminates in finite time. Let $s = \gamma_0\gamma_1 \dots \gamma_{l-1}$ be the smallest prefix of c of length l that contains distinguishable configurations only. Recall that we assume that no two consecutive configurations of c are identical. Since s is the smallest prefix of c containing only distinguished configurations by assumption there exists γ_i in s such that γ_i and γ_l are undistinguished. Such a γ_l exists since we assume that c contains at least two undistinguished configurations. So, executing $\mathcal{P}$, the set of actions that led the system from γ_i to γ_{i+1} are also applicable in γ_l. Thus, by executing the same set of actions, there exists a configuration γ_{l+1} reachable from γ_l such that γ_{i+1} and γ_{l+1} are undistinguished. Following the same reasoning for every $j \in [2..l-1]$, there exists a configuration γ_{l+j} reachable from γ_{l+j-1} such that γ_{i+j} and γ_{l+j} are undistinguished. Therefore, there exists an execution c of $\mathcal{P}$ leading to the infinite sequence of actions starting for γ_i. This contradicts the assumption that c terminates in finite time. □

3 Visibility $\phi = 1$

In this section, we first prove that no deterministic exploration is possible in the semi-asynchronous model when robots are able to see at distance 1 only ($\phi = 1$). The result holds for any $k < n$. We then show that no deterministic exploration solves the problem with four robots, even in the (fully) synchronous model when $n > 6$. The above results are also valid for the asynchronous model [16]. Next, we provide optimal deterministic algorithms in the synchronous model for both cases $3 \leq n \leq 6$ and $n > 6$.

3.1 Negative Results

Asynchronous Model. Since robots are able to see only at distance 1, only the four following rules are possible:

$\mathcal{R}_{\texttt{sgl}}$ $0(1)0 :: \rightarrow \vee \leftarrow$
$\mathcal{R}_{\texttt{out}}$ $0(1)1 :: \leftarrow$
$\mathcal{R}_{\texttt{in}}$ $0(1)1 :: \rightarrow$
$\mathcal{R}_{\texttt{swp}}$ $1(1)1 :: \rightarrow \vee \leftarrow$

In the following, γ_0 refers to an initial configuration. Let us first assume that γ_0 consists of a single 1.block of size k. Note that Rule $\mathcal{R}_{\texttt{sgl}}$ is not applicable in an initial configuration. Also, Rules $\mathcal{R}_{\texttt{out}}$ and $\mathcal{R}_{\texttt{in}}$ (Rule $\mathcal{R}_{\texttt{swp}}$) implies that k must be greater than 2 (3, respectively). We first show that no deterministic exploration protocol includes Rules $\mathcal{R}_{\texttt{sgl}}$, $\mathcal{R}_{\texttt{out}}$, and $\mathcal{R}_{\texttt{swp}}$ starting from γ_0. We now show that Rules $\mathcal{R}_{\texttt{out}}$ and $\mathcal{R}_{\texttt{swp}}$ cannot be part of the protocol.

Lemma 3.1 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then $\mathcal{P}$ does not include Rule $\mathcal{R}_{\texttt{out}}$.*

Proof. By contradiction, assume that $\mathcal{P}$ includes Rule $\mathcal{R}_{\texttt{out}}$. Note that Rule $\mathcal{R}_{\texttt{out}}$ is only enabled on robots that are at the border of a 1.block. Since there is only one ϕ-group in γ_0 (the initial configuration), there are only two robots that can execute Rule $\mathcal{R}_{\texttt{out}}$. Let $u_iu_{i+1} \dots u_{i+k}$ be the 1.block in γ_0. Denote r_j the robot located on node u_{i+j} ($0 \leq i < k$). Without lost of generality, assume that the adversary activates r_0 only in γ_0. Let us call the resulting configuration by T (standing for "*Trap*" configuration). There are two cases to consider:

- $k = n - 1$. In that case, r_0 moves to u_{i+k+1} and T includes a 1.block $u_{i+1} \dots u_{i+k}u_{i+k+1}$. T is undistinguishable from γ_0. This contradicts Theorem 2.1.

- $k < n - 1$. In that case, once r_0 moves, it becomes an isolated robot on u_{i-1} in T. Again, there are two cases. Assume first that $k = 2$. Then, T includes two isolated robots, r_0 and r_1. Even if $\mathcal{P}$ includes Rule $\mathcal{R}_{\texttt{sgl}}$, by activating r_0 and r_1 simultaneously, $T + 1$ is undistinguishable from T. Again, this contradicts Theorem 2.1. Assume that $k > 2$. Then, T is the configuration in which u_{i+1} is the border occupied by r_1 of the 1.block $u_{i+1} \dots u_{i+k}$. Assume that in T, the adversary activates r_1 that executes Rule $\mathcal{R}_{\texttt{out}}$. Then, Configuration $T + 1$ includes two 1.blocks disjoint by one node. The former includes r_0 and r_1 (on u_{i-1} and u_i, respectively). The latter forms the sequence $r_2 \dots r_k$, located on $u_{i+2} \dots u_{i+k}$. More generally, assume that $\forall i \in [1..k-2]$, in $T + i$, the adversary activates r_{i+1}. The resulting configuration $T + k - 2$ is undistinguishable from T. A contradiction (Theorem 2.1).

□

Lemma 3.2 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then Rule $\mathcal{R}_{\texttt{in}}$ and Rule $\mathcal{R}_{\texttt{swp}}$ are mutually exclusive with respect to $\mathcal{P}$.*

Proof. Assume by contradiction that $\mathcal{P}$ includes both Rule $\mathcal{R}_{\texttt{in}}$ and Rule $\mathcal{R}_{\texttt{swp}}$. Note that Rule $\mathcal{R}_{\texttt{in}}$ (respectively, Rule $\mathcal{R}_{\texttt{swp}}$) is enabled only on robots that are located at the border of the 1.block (resp., inside the 1.block). In the case where k is even, assume that the scheduler activates all the robots. The resulting configuration is undistinguishable from γ_0. A contradiction (by Theorem 2.1). In the case where k is odd, suppose that the scheduler activates the robots at the border of the 1.block and their neighbors (in the case where $k = 3$, only one extremity is activated). The resulting configuration is undistinguishable from γ_0. A contradiction (by Theorem 2.1). □

Lemma 3.3 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then $\mathcal{P}$ does not include Rule $\mathcal{R}_{\texttt{swp}}$.*

Proof. By contradiction, assume that $\mathcal{P}$ includes Rule $\mathcal{R}_{\texttt{swp}}$. From Lemmas 3.1 and 3.2, $\mathcal{P}$ does not include Rules $\mathcal{R}_{\texttt{out}}$ nor Rule $\mathcal{R}_{\texttt{in}}$. So, in γ_0, the robots that are inside a 1.block are the only robots that are able to execute an action in the initial configuration (and k must be greater than or equal to 3). If $k > 3$, then, the adversary activates at each time two neighboring robots such that once they move, they simply exchange their position and the resulting configuration is undistinguishable from γ_0. A contradiction.
If $k = 3$ then, when Rule $\mathcal{R}_{\texttt{swp}}$ is executed, a 2.tower is created. Rule 0(2)0:: $\leftarrow \vee \rightarrow$ cannot be enabled since once executed, the scheduler can activate only one robot in the 2.tower such that the configuration reached is undistinguishable from γ_0. In the case where $\mathcal{R}_{\texttt{sgl}}$ is enabled, then the isolated robot becomes either (i) neighbor of the 2.tower (the case where $n = 4$) or (ii) remains isolated. In the first case (i), if either 0(1)2:: $\leftarrow$ or 0(2)1:: $\rightarrow$ is enabled, a 3.tower is created (Observe that one node of the ring has not been explored). The only rule that can be executed is 0(3)0:: $\leftarrow \vee \rightarrow$. Suppose that the scheduler activates two robots that move in two opposite directions. The configuration reached is undistinguishable from γ_0. If 0(2)1:: $\leftarrow$ is enabled, then suppose that the scheduler activates only one robot. The configuration reached is undistinguishable from γ_0. In the second case (ii) (the isolated robot remains isolated), $\mathcal{R}_{\texttt{sgl}}$ keeps being enabled.

Once it it is executed, the isolated robot moves back to its previous position and the configuration reached is indistinguishable from the previous one. □

Corollary 3.1 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$, $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then $\mathcal{P}$ includes Rule $\mathcal{R}_{\texttt{in}}$ that is the only applicable rule in the initial configuration.*

Proof. Directly follows from Lemmas 3.1, 3.2, and 3.3. □

Lemma 3.4 *Let $\mathcal{P}$ be a semi-synchronous exploration protocol for $\phi = 1$, $2 \leq k < n$. Then, k must be greater than or equal to 5.*

Proof. Assume that the adversary activates both robots r_0 and r_1 at the border of the 1.block in γ_0. By Corollary 3.1, both r_0 and r_1 execute Rule $\mathcal{R}_{\texttt{in}}$ in γ_0.

If $k = 2$, the system reaches a configuration γ_1 that is undistinguishable from γ_0.

If $k = 3$, then a 3.tower is created on the middle node of the 1.block in γ_1. Since the ring is not explored anymore (at least one node is not visited), $\mathcal{P}$ must include at least the following rule: $0(3)0 :: \leftarrow \vee \rightarrow$. Then, the adversary activates only two of them and an opposite destination node for each of them. Then, the system reaches a configuration γ_2 that is undistinguishable from γ_0.

If $k = 4$, then two neighboring 2.towers are created on the two middle nodes of the 1.block in γ_1. There are two cases to consider: Assume first that $\mathcal{P}$ includes the rule: $2(2)0 :: \leftarrow$. Then, by activating the same number of robots on each tower, the system reaches γ_2 such that γ_1 and γ_2 are two undistinguishable configurations. Assume that $\mathcal{P}$ includes the rule: $2(2)0 :: \rightarrow$. Again, by activating only one robot on each tower, the system reaches a configuration γ_2 that is undistinguishable from γ_0. □

Let us call a *symmetric x.tower sequence* (an S^x-sequence for short), a sequence of occupied nodes such that the extremities of the sequence contain an x.tower. Observe that a tower containing at least 2 robots (by definition), x is greater than or equal to 2. Also, since the robots can only see at distance 1, the tower is only seen by its neighboring robots.

The following lemma directly follows from Corollary 3.1 and Lemma 3.4:

Lemma 3.5 *For every semi-synchronous exploration protocol $\mathcal{P}$ for $\phi = 1$, then $5 \leq k < n$ and there exists some executions of $\mathcal{P}$ leading to a configuration containing an S^2-sequence.*

Consider robots being located at the border of an 1.block. Let $x \geq 1$ be the number of robots located on the border node and the following generic rules:

$\mathcal{T}\alpha(x)$ $\quad 0(x)1 :: \leftarrow$
$\mathcal{T}\beta(x)$ $\quad 0(x)1 :: \rightarrow$
$\mathcal{T}\gamma(x)$ $\quad x(1)1 :: \leftarrow$
$\mathcal{T}\delta(x)$ $\quad x(1)1 :: \rightarrow$

Remark that Rule $\mathcal{R}_{\texttt{in}}$ corresponds to Rule $\mathcal{T}\beta(1)$. Also, note that since x robots are located on the border node of an 1.block, the local configuration for both $\mathcal{T}\gamma(x)$ and $\mathcal{T}\delta(x)$ is $0x11$. Similarly, define the generic rule $\mathcal{T}_{\texttt{sgl}}(y)$ $(y \geq 2)$ as follow: $0(y)0 ::\leftarrow \vee \rightarrow$.

Lemma 3.6 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $5 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then for every $x \geq 2$, Rule $\mathcal{T}_{\texttt{sgl}}(x)$ is mutually exclusive with both $\mathcal{T}\gamma(x-1)$ and $\mathcal{T}\delta(x)$, with respect to $\mathcal{P}$.*

Proof. Assume that $\mathcal{P}$ includes $\mathcal{T}_{\mathtt{sgl}}(x)$ for some $x \geq 2$. There are two cases to consider: Assume by contradiction that $\mathcal{P}$ includes either Rule $\mathcal{T}\gamma(x-1)$ or Rule $\mathcal{T}\delta(x)$. Then, starting from a configuration $0(x-1)11$ or $0(x)11$, $\mathcal{T}\gamma(x-1)$ or $\mathcal{T}\delta(x)$ is enabled, respectively. In both cases, by executing $\mathcal{T}\gamma(x-1)$ or $\mathcal{T}\delta(x)$, the configuration locally becomes $0x01$ in which Rule $\mathcal{T}_{\mathtt{sgl}}(x)$ is enabled for each robot belonging to the x.tower. By executing $\mathcal{T}_{\mathtt{sgl}}(x)$ on one robot only, the adversary can lead the system in a configuration that is undistinguishable from the previous one. A contradiction by Theorem 2.1. □

Lemma 3.7 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $5 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then for every odd $x \geq 3$, the rules $\mathcal{T}_{\mathtt{sgl}}(x)$ and $\mathcal{T}\beta(\lfloor \frac{x}{2} \rfloor)$ are mutually exclusive with respect to $\mathcal{P}$.*

Proof. Assume by contradiction that $\mathcal{P}$ includes both $\mathcal{T}_{\mathtt{sgl}}(x)$ and $\mathcal{T}\beta(\lfloor \frac{x}{2} \rfloor)$ for some odd $x \geq 3$. Starting from a configuration in which $\mathcal{T}_{\mathtt{sgl}}(x)$ is enabled, by executing $\mathcal{T}_{\mathtt{sgl}}(x)$, the adversary can execute $\mathcal{T}_{\mathtt{sgl}}(x)$ on $x-1$ robots, moving the half of them on the right side and the half of them on the left side. By doing that, the system reaches either a towerless 1.block or an $S^{\lfloor \frac{x}{2} \rfloor}$-sequence, depending on whether $x = 3$ or $x \geq 5$, respectively. In both cases, $\mathcal{T}\beta(\lfloor \frac{x}{2} \rfloor)$ is enabled on each robot belonging to an extremity of the 1.block (if $x = 3$, then $\mathcal{T}\beta(\lfloor \frac{x}{2} \rfloor)$ corresponds to Rule $\mathcal{R}_{\mathtt{in}}$). By executing $\mathcal{T}\beta(\lfloor \frac{x}{2} \rfloor)$ on them, the adversary can bring the system in a configuration that is undistinguishable from the previous one. A contradiction (Theorem 2.1). □

Lemma 3.8 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $5 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then for every $x \geq 1$, $\mathcal{P}$ includes $\mathcal{T}\beta(x)$ only.*

Proof. The lemma is proven for $x = 1$ by Corollary 3.1. Assume by contradiction that there exists $x \geq 2$ such that $\mathcal{P}$ does not include Rule $\mathcal{T}\beta(x)$ only. From Lemma 3.5, there exists some executions of $\mathcal{P}$ leading to a configuration containing an S^2-sequence. Let us consider the smallest x satisfying the above condition. So, starting from an S^2-sequence, for every $y \in [1, x-1]$, by activating Rule $\mathcal{T}\beta(y)$ on every robot at the extremity of the S^y-sequence, in $x-1$ steps, the adversary can lead the system into a configuration γ_{x-1} containing an S^x-sequence. Since $\mathcal{P}$ does not include only Rule $\mathcal{T}\beta(x)$, there are two cases:

1. $\mathcal{P}$ includes Rule $\mathcal{T}\alpha(x)$. In that case, by activating $x-1$ robots on each x.tower of the S^x-sequence, the system reaches an S^{x-1}-sequence. This configuration is undistinguishable with the configuration γ_{x-2}, the previous configuration of γ_{x-1} in the execution.

2. $\mathcal{P}$ does not include Rule $\mathcal{T}\alpha(x)$. Then, the neighbors of the towers are the only robots that can be activated with either $\mathcal{T}\gamma(x)$ or $\mathcal{T}\delta(x)$. In the first case ($\mathcal{T}\gamma(x)$ is enabled), the system reaches a configuration similar to the one shown in Figure 3.1, *i.e.*, a 1.block surrounded by two isolated $x+1$.towers. The size of the 1.block is equal to $k - 2(x+1)$. By Lemma 3.6, $\mathcal{P}$ does not include $\mathcal{T}_{\mathtt{sgl}}(x+1)$. So, none of the robots belonging to a tower is enabled. Furthermore, the local view of both extremities of the 1-block being similar to the initial configuration γ_0, from Corollary 3.1, both can be activated with Rule $\mathcal{R}_{\mathtt{in}}$ by the adversary. Then, the system reaches a configuration similar to the one shown in Figure 3.2, *i.e.*, an S^2-sequence, surrounded with 2 towers and separated by one single empty node. In the second case ($\mathcal{T}\delta(x)$ is enabled), the system reaches a configuration that is also similar to the one shown in Figure 3.2. Following

the same reasoning as above, the system reaches a configuration similar to the one shown in Figures (3.3, 3.4) or in Figures (3.5, 3.6) depending on whether k is odd or even.

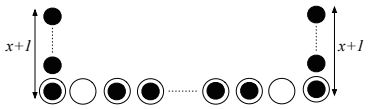


Figure 3.1: $\mathcal{T}\gamma(x)$ is executed.

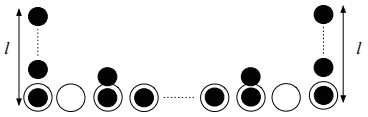


Figure 3.2: $\mathcal{T}\delta(x)$ is executed

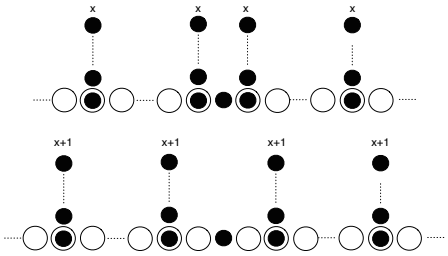


Figure 3.3: The case where $\kappa = 1$

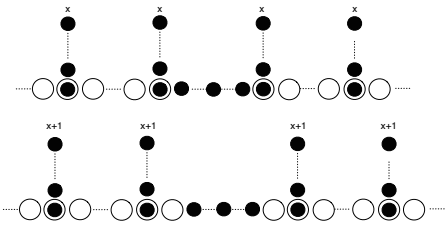


Figure 3.4: The case where κ is odd and $\kappa \geq 3$

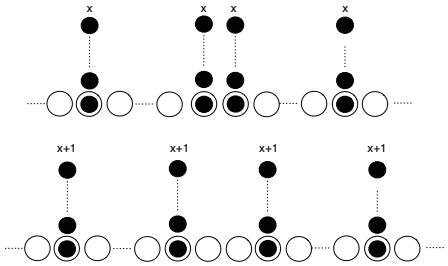


Figure 3.5: The case where $\kappa = 0$

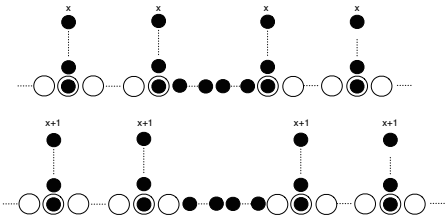


Figure 3.6: The case where κ is even and $\kappa \geq 2$

Let τ, S, and κ be the number of isolated towers, their size, and the number of robots surrounded by τ towers, respectively. By construction, $\tau = \lfloor \frac{k}{S} \rfloor$, $\kappa = k \bmod (\tau S)$, and $\kappa \leq 2x$. There are two cases to consider:

(a) κ is odd. Again consider two cases:

i. $\kappa \geq 3$. Then, $\lfloor \frac{\kappa}{2} \rfloor < x$ and by assumption, for every $y \in [1, \lfloor \frac{\kappa}{2} \rfloor]$, $\mathcal{P}$ includes the rules $\mathcal{T}\beta(y)$. So, the adversary can lead the system into a configuration containing a single $\kappa+1$.tower surrounded with τ towers. Since $\frac{\kappa}{2} < x$ and since by assumption, for every $y \in [2, \frac{\kappa}{2}\rfloor]$, $\mathcal{P}$ includes Rule $\mathcal{T}\beta(y)$, by Lemma 3.7, $\mathcal{P}$ does not include Rule $\mathcal{T}_{\mathtt{sgl}}(\kappa)$. So, no robot of the κ.tower is enabled. So, the system is at a deadlock and some nodes are not visited by at least one robot. This contradicts that $\mathcal{P}$ is an exploration protocol.

ii. $\kappa = 1$. The only applicable rules on the single robot is either (i) Rule $\mathcal{R}_{\mathtt{sgl}}$ or (ii) Rule $\mathcal{R}_{\mathtt{sp}}$ defined as follow: $x(1)x ::\leftarrow \vee \rightarrow$ depending on whether $\mathcal{P}$ includes $\mathcal{T}\gamma(x)$ or $\mathcal{T}\delta(x)$, respectively. In the first case (i), after executing the rule, the single robot is located on the neighboring node of one of the towers, *i.e.*, a configuration containing $0(y)10$ where y is equal to $x+1$. The only possible action is then to create

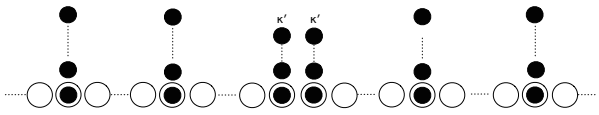


Figure 3.7: The reached configuration when $\kappa > 2$.

an $y+1$.tower. If $\mathcal{P}$ includes $\mathcal{T}_{\texttt{sgl}}(y+1)$, then by selecting a single robot of the $y+1$.tower, the adversary can lead the system into the previous configuration. A contradiction (Theorem 2.1). In the second case (ii), an $x+1$.tower is created. As in Case (i), if $\mathcal{P}$ includes $\mathcal{T}_{\texttt{sgl}}(x+1)$, then by selecting a single robot of the $x+1$.tower, the adversary can lead the system into the previous configuration. A contradiction (Theorem 2.1).

(b) κ is even. There are three cases.

i. $\kappa = 0$. Since none of the robots of τ towers is enabled, the system is at a deadlock. This contradicts that $\mathcal{P}$ is an exploration protocol.

ii. $\kappa = 2$. in this case there is a single 1.block of size 2 having either two neighboring x.towers at distance 1 or $x+1$.towers at distance 2 depending on whether $\mathcal{P}$ includes $\mathcal{T}\gamma(x)$ or $\mathcal{T}\delta(x)$, respectively. In both cases, the only applicable rule is $\mathcal{T}\delta(x)$. Assume that the scheduler activates both robots at the same time. Then, the system remains into an undistinguishable configuration. A contradiction.

iii. $\kappa > 2$. Then, $\frac{\kappa}{2} < x$ and by assumption, for every $y \in [2, \frac{\kappa}{2}\rfloor]$, $\mathcal{P}$ includes the rules $\mathcal{T}\beta(y)$, the adversary leads the system into a configuration containing two neighboring κ'.towers, $\kappa' = \frac{\kappa}{2}$—refer to Figure 3.7. Assume that $\mathcal{P}$ includes Rule $\mathcal{T}_{\texttt{dbl}}(\kappa')$ defined as follows: $0(\kappa')\kappa' ::\leftarrow$. Then by choosing to execute Rule $\mathcal{T}_{\texttt{dbl}}(\kappa')$ on $\kappa'-1$ robots on each tower (since Rule $\mathcal{R}_{\texttt{out}}$ cannot be a part of $\mathcal{P}$, $\kappa > 2$), the adversary brings the system into an $S^{\kappa'-1}$-sequence that is undistinguisable from the previous configuration. If Rule $\mathcal{T}_{\texttt{dbl}}(\kappa')$ is defined as follows: $0(\kappa')\kappa' ::\rightarrow$. Then by choosing to execute Rule $\mathcal{T}_{\texttt{dbl}}(\kappa')$ on the same number of robots on each κ'.tower, the system remains into an undistinguishable configuration. In both cases, this contradicts Theorem 2.1.

□

Let us now suppose that γ_0 is any arbitrary starting configuration. The following lemma holds:

Lemma 3.9 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then Rule $\mathcal{R}_{\texttt{out}}$ and Rule $\mathcal{R}_{\texttt{sgl}}$ are mutually exclusive with respect to $\mathcal{P}$.*

Proof. Assume by contradiction that $\mathcal{P}$ includes both Rule $\mathcal{R}_{\texttt{out}}$ and Rule $\mathcal{R}_{\texttt{sgl}}$. Note that Rule $\mathcal{R}_{\texttt{out}}$ (respectively, Rule $\mathcal{R}_{\texttt{sgl}}$) is enabled only on robots that are located at the border of an 1.block (resp., on isolated robots). Suppose that the initial configuration γ_0 contains two isolated robots, r_1 and r_2, that is at distance 2 from a 1.block, one at each side. By executing $\mathcal{R}_{\texttt{sgl}}$ on both r_1 and r_2, the adversary can bring the system in a configuration γ_1 where both r_1 and r_2 are located at the border of the 1.block. Then, Rule $\mathcal{R}_{\texttt{out}}$ becomes enabled on both r_1 and r_2. By executing $\mathcal{R}_{\texttt{out}}$ on both r_1 and r_2, the adversary brings the system in a configuration γ_2 that is undistinguishable from γ_0. A contradiction. □

Lemma 3.10 *Let $\mathcal{P}$ be a semi-synchronous protocol for $\phi = 1$ and $2 \leq k < n$. If $\mathcal{P}$ solves the exploration problem, then $\mathcal{P}$ does not include Rule $\mathcal{R}_{\texttt{out}}$.*

Proof. By contradiction, assume that $\mathcal{P}$ includes Rule $\mathcal{R}_{\texttt{out}}$. Note that Rule $\mathcal{R}_{\texttt{out}}$ is only enabled on robots that are at the border of a 1.block. Suppose that γ_0 contains two 1.blocks at distance 2 from each other. Assume that the scheduler activates only one extremity of one of the two 1.blocks. Let us refer to the 1.block from which one extremity is activated by $B1$ and let refer to the other 1.block by $B2$. Let $u_i u_{i+1} \dots u_{i+m}$ be the 1.block $B1$ in γ_0 and let $u_{i-2} u_{i-3} \dots u_{i-m'}$ be the 1.block $B2$ (m and m' are the size of respectively $B1$ and $B2$). Denote r_j the robot located on node u_{i+j} ($0 \leq i < k$). Without lost of generality, assume that the adversary activates r_0 only in γ_0. Once r_0 moves, it joins $B2$ in γ_1. Assume that in γ_1 the scheduler activates r_1 that executes $\mathcal{R}_{\texttt{out}}$ again. More generally, assume that $\forall\ i \in [1, \dots, m-2]$, in γ_{1+i}, the adversary activates r_{i+1}. Once the configuration γ_{k-1} is reached, the scheduler activates the same robots in the reverse order (note that these robots are part of $B2$). At each time, one robot is activated, it joins back $B1$. When $m-2$ robots are activated, the configuration reached is undistinguishable from γ_1. A contradiction. □

Observe that both Lemmas 3.2 and 3.3 are valid when γ_0 is any towerless configuration. In the case where there are isolated robots that are at distance 2 from a 1.block, the scheduler activates them, such that they join a 1.block by executing $\mathcal{R}_{\texttt{sgl}}$. The only rule that can be enabled is $\mathcal{R}_{\texttt{in}}$. Each 1.block, behaves independently from the other 1.blocks. Lemmas 3.4 to 3.8 are also valid. Therefore:

Theorem 3.1 *No deterministic exploration protocol exists in the semi-synchronous model (ATOM) for $\phi = 1$, $n > 1$, and $1 \leq k < n$.*

Proof. The theorem is a direct consequence of Lemma 3.4 and Lemma 3.8 and its proof. □

Since the set of executions that are possible in CORDA is a strict superset of those that are possible in ATOM, Theorem 3.1 also holds in the CORDA [16].

Corollary 3.2 *No deterministic exploration protocol exists in the asynchronous model (CORDA) for $\phi = 1$, $n > 1$, and $1 \leq k < n$.*

Synchronous Model.

Theorem 3.2 *Let $\mathcal{P}$ be a synchronous exploration protocol for $\phi = 1$ and $2 \leq k < n$. If $n > 7$, then, k must be greater than or equal to 5.*

Proof.

The proof is by contradiction. Assume that there exists a synchronous exploration protocol $\mathcal{P}$ for $\phi = 1$, $2 \leq k < 5$, and $n > 7$.

Let us start from a possible starting configuration, the idea is to derive all the possible executions and then show that non of them can ensure the completion of the exploration task. The cases bellow are possible:

1. $k = 1$. It is clear that no exploration is possible by a single robot on the ring.

2. $k = 2$. The robots in this case, either move towards each other exchanging their position or, they move in the opposite direction and become isolated robots. In the latter case, once they are activated again, the scheduler brings them back to their previous position. The configuration reached is in both cases is *undistinguishable* from the starting one. A contradiction.

3. $k = 3$. (i) In the case where only Rule $\mathcal{R}_{\texttt{out}}$ is enabled. Once the rule is executed, all robots in the configuration become isolated robots. Thus no exploration is possible (all the configurations that are created after are undistinguishable). (ii) If Rule $\mathcal{R}_{\texttt{in}}$ is enabled alone, then the configuration reached contains a single 3.tower. All robots part of the 3.tower behave as a single robot. According to Case (2), no exploration is possible in this case. (iii) Rule $\mathcal{R}_{\texttt{swp}}$, is the only one enabled. In this case the configuration reached contains an isolated robot and an isolated 2.tower (let refer to this configuration by T). Note that only robots part of the tower are enabled. Suppose that the scheduler activates them such that they move in the opposite direction of the isolated robots, and then, it activates them again but this time, they move in the opposite direction (towards the isolated robot). The configuration reached is undistinguishable from T configuration. Thus no exploration is possible. (iv) If Rules $\mathcal{R}_{\texttt{swp}}$ and $\mathcal{R}_{\texttt{in}}$ are enabled at the same time, each node of the ring is visited by robots by executing the following two rules: $0(2)1 ::\rightarrow$ and $2(1)0 ::\rightarrow$, however, robots are not able to detect the end of the exploration. Thus, no exploration is possible in this case. Finally, if (v) Rules $\mathcal{R}_{\texttt{swp}}$ and $\mathcal{R}_{\texttt{out}}$ are enabled at the same time. Once such rules are executed, the configuration reached contains one 1.block of size 2 and one isolated robots. By executing Rule $\mathcal{R}_{\texttt{out}}$ (possibly $\mathcal{R}_{\texttt{sgl}}$), a configuration with just isolated robots is eventually reached. Thus, no exploration is possible in this case too.

4. $k = 4$. Suppose that the initial configuration contains a single 1.block of size 4. The cases bellow are possible:

   (a) Rule $\mathcal{R}_{\texttt{swp}}$ is executed. The two robots inside the 1.block exchange their position and the configuration reached is *undistinguishable* from the starting one. A contradiction.

   (b) Rule $\mathcal{R}_{\texttt{in}}$ is executed. Two neighboring towers are then created. If the towers move towards each other, the configuration reached is *undistinguishable* from the starting one. If they move in the opposite direction of each other, then they become both isolated towers *i.e.*, the robots in the tower cannot see any neighboring robot. When they are activated again, the scheduler brings them back to their previous position. Thus, in this case too, the configuration reached is *undistinguishable* from the starting one. A contradiction.

   (c) Rule $\mathcal{R}_{\texttt{out}}$ is executed. The robots that were at the extremity of the 1.block become isolated robots. if Rule $\mathcal{R}_{\texttt{sgl}}$ is enabled then two isolated towers are created (Note that there is at least one node that has not been explored yet). Robots in the same tower behaves as a single robot. Thus, the configuration is similar to the one containing only two robots. According to Case (2), no exploration is possible in this case. Hence, Rule $\mathcal{R}_{\texttt{sgl}}$ cannot be enabled. Rule $\mathcal{R}_{\texttt{out}}$ is the only one that is enabled on robots part of the 1.block of size 2. Once they move, two 1.blocks of size 2 are created. All the robots have now the same view. Note that Rule $\mathcal{R}_{\texttt{out}}$ remains the only one enabled on all the robots. Once it is executed, a new 1.block of size 2 is created and the configuration contains either (i) another 1.block of size 2 ($n = 8$) or (ii) a tower ($n = 7$) or (iii) two isolated robots ($n > 8$). In the first case (i), $\mathcal{R}_{\texttt{out}}$ remains enabled on all the robots. Once it is executed, the configuration reached is undistinguishable from one of the previous ones.

Thus, no exploration is possible. In the second case (*ii*), $\mathcal{R}_{\texttt{out}}$ remains enabled on robots part of the 1.block. Observe that the robots in the tower cannot be activated otherwise, an undistinguishable configuration is reached (the one that contains two 1.blocks). Thus only $\mathcal{R}_{\texttt{out}}$ is enabled. Once it is executed, the configuration contains two isolated robots. Such isolated robots can never detect the end of the exploration task since no rule can be executed anymore. Thus no exploration is possible in this case too. In the third case (*iii*) on robots part of the 1.block, Rule $\mathcal{R}_{\texttt{out}}$ is enabled, once they move all the robots become isolated robots. Hence, the scheduler can choose always the same direction for all the robots. The configuration reached at each time is then *undistinguishable* from the previous one. A contradiction.

(d) All the robots are allowed to move. Note that in this case either two towers are created and it is clear that no exploration is possible (since each tower behaves as a single robot and thus, the system behaves as there is two robots on the ring). or we retrieve case 4c. Thus, no exploration is possible in this case too.

From the cases above, we can deduce that no exploration is possible for $n > 7$ using four robots even in the fully-synchronous model. □

3.2 Synchronous Algorithms

In the following, we present optimal deterministic algorithms that solve the exploration problem in the fully-synchronous model for any ring $3 \leq n \leq 6$ and $n \geq 7$, using respectively $(n-1)$ robots (except for the case $n = 6$ that needs only 4 robots) and five robots. Note that the starting configuration contains a single 1.block (refer to Figure 3.8).

Fully-Synchronous Exploration for $k = 5$ and $n \geq 7$. The idea of Algorithm 1 is as follow: The robots that are at the border of the 1.block are the only ones that are allowed to move in the initial configuration, γ_0. Their destination is their adjacent occupied node (Rule 1A1). Since the system is synchronous, the next configuration, γ_1, contains a single robot surrounded by two 2.towers. In the next step, the towers move in the opposite direction of the single robot (Rule 1A3) and the single robot moves towards one of the two towers (Rule 1A2). Note that the resulting configuration γ_2 is 21020^{n-4}, providing an orientation of the ring. From there, the single 2.tower are the landmark allowing to detect termination and the three other robots explore the ring by perform Rules 1A3 and 2A4 synchronously. After $n-4$ steps, $n-4$ nodes are visited by the 3 robots and the system reaches γ_{n-2} that is equal to 2210^{n-3}. Finally, by performing Rule 2A4, the single robot create a 3.tower, marking the end of the exploration in γ_{n-1} in which each robot is awake of the termination.

Algorithm 1 Fully-Synchronous Exploration for $n \geq 7$ (visibility 1)

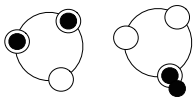
1A1:	0(1)1	::	$\rightarrow$	*// Move towards my occupied neighboring node*
1A2:	2(1)2	::	$\rightarrow \vee \leftarrow$	*// Move towards one of my neighboring node*
1A3:	0(2)1	::	$\leftarrow$	*// Move towards my neighboring empty node*
2A4:	2(1)0	::	$\leftarrow$	*// Move to the tower*

Fully-Synchronous Exploration for $3 \leq n \leq 6$. The formal description of the algorithm is given in Algorithm 2. The robots in this case detect the end of the exploration task if they are

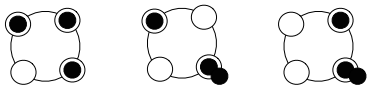
either part of a 2.tower or neighbors of a 2.tower. The idea of the algorithm is the following: For $3 \leq n \leq 5$, $k = (n-1)$ robots are necessary to perform the exploration task. The robots that are at the extremities of the 1.block are the ones allowed to move, since $k = n - 1$, once they move, a 2.tower is created. If the reached configuration contains an isolated robot (the case where $n = 4$) then, this robot is the only one allowed to move. Its destination is one of its adjacent empty nodes (refer to Rule $1A'2$). Once it moves it becomes neighbor of the 2.tower. In the case where ($n = 6$), 4 robots are necessary to solve the exploration problem. In the same manner, the robots that are at the border of the 1.block are the ones allowed to move, their destination is their adjacent empty node (refer to Rule $1A'1$). Once they move, the configuration reached contains two 1.blocks of size 2. Observe that on all the robots Rule $1A'1$ is enabled. Once it is executed, two 2.towers are created. The robots can detect the end of the exploration since they are all part of a 2.tower.

Algorithm 2 Fully-Synchronous Exploration for $3 \leq n \leq 6$ (visibility 1)

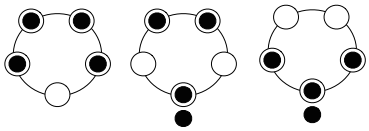
1A'1:	0(1)1	::	$\leftarrow$	*// Move towards my neighboring empty node*
1A'2:	0(1)0	::	$\leftarrow \vee \rightarrow$	*// Move towards one of my neighboring nodes*



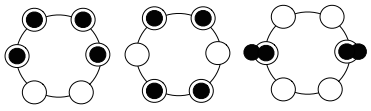
(a) Fully-synchronous exploration $n = 3$ and $k = 2$



(b) Fully-synchronous exploration $n = 4$ and $k = 3$



(c) Fully-synchronous exploration $n = 5$ and $k = 4$



(d) Fully-synchronous exploration $n = 6$ and $k = 4$

Figure 3.8: Fully-Synchronous Deterministic Solutions for $3 \leq n \leq 6$.

4 Visibility $\phi = 2$

In this section we present two deterministic algorithms that solve the exploration problem. The first one uses 9 robots and works for any towerless initial configurations that contains a single ϕ.group. The second one uses 7 robots but works only when the starting configuration contains a single 1.block of size 7. In both algorithms we suppose that $n \geq \phi k + 1$.

4.1 Asynchronous Exploration using 9 robots

Let us first define two special configurations:

Definition 4.1 *A configuration is called* Middle *(refer to Figure 4.9) at instant t if there exists a sequence of consecutive nodes $u_i, u_{i+1}, \ldots, u_{i+5}, u_{i+6}$ such that:*

Figure 4.9: *Middle* configuration

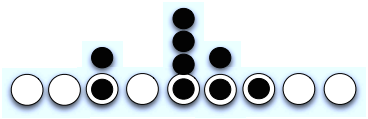


Figure 4.10: *Terminal* configuration

- $M_j = 2$ *for* $j \in \{i, i+1, i+5, i+6\}$
- $M_j = 1$ *for* $j = i+2$
- $M_j = 0$ *for* $j \in \{i+3, i+4\}$

Definition 4.2 *A configuration is said* Terminal *(refer to Figure 4.10) at instant t if there exists a sequence of nodes* $u_i, u_{i+1}, \ldots, u_{i+4}$ *such that:*

- $M_j = 2$ *for* $j \in \{i, i+3\}$
- $M_j = 4$ *for* $j = i+2$
- $M_j = 0$ *for* $j = i+1$
- $M_j = 1$ *for* $j = i+4$

The algorithm comprises two phases as follow:

1. `Organization Phase`. The aim of this phase is to build the `Middle Configuration`. The initial configuration is any towerless configuration that contains a single ϕ.group.
2. `Exploration Phase`. The starting configuration of this phase is the `Middle Configuration`. A set of robots are elected to perform the exploration while other robots stay still as a land mark. `Terminal` configuration is created at this end of this phase to indicate the end of exploration task.

The formal description of the algorithm is presented in Algorithms 3.

Algorithm 3 Asynchronous Exploration using 9 robots ($\phi = 2$)

`Organization` Phase				
2A'1:	00(1)01	::	$\rightarrow$	*// Move toward the occupied node*
2A'2:	00(1)1?	::	$\rightarrow$	*// Move to my adjacent occupied node*
2A'3:	00(2)01	::	$\rightarrow$	*// Move toward the occupied node*
2A'4:	02(1)01	::	$\rightarrow$	*// Move in the opposite direction of the tower*
2A'5:	21(1)01	::	$\rightarrow$	*// Move in the opposite direction of the tower*
2A'6:	21(1)1?	::	$\leftarrow$	*// Move toward the tower*
2A'7:	20(1)02	::	$\leftarrow \vee \rightarrow$	*// Move to one of my neighboring nodes*
`Exploration` Phase				
2A'6:	21(1)1?	::	$\leftarrow$	*// Move toward the tower*
2A'8:	00(2)21	::	$\leftarrow$	*// Move in the opposite direction of my neighboring tower*
2A'9:	01(1)21	::	$\leftarrow$	*// Move in the opposite direction of the tower*
2A'10:	20(2)10	::	$\leftarrow$	*// Move toward the tower at distance* 2
2A'11:	20(1)00	::	$\leftarrow$	*// Move toward the tower*
2A'12:	20(2)21	::	$\leftarrow$	*// Move to my adjacent free node*
2A'13:	21(1)21	::	$\leftarrow$	*// Move in the opposite direction of my neighboring tower*
2A'14:	02(2)22	::	$\rightarrow$	*// Move toward the tower having another tower as a neighbor*
2A'15:	02(1)32	::	$\rightarrow$	*// Move toward the* 3*.tower*

Proof of correctness We prove in this section the correctness of our algorithm presented above. In the following, a configuration is called `intermediate` if Rule $2A'7$ is enabled in the configuration.

Lemma 4.1 *Starting from any towerless configuration that contains a single $\phi.group$,* `intermediate` *configuration is eventually reached.*

Proof. Two cases are possible as follow:

- The initial configuration contains a 1.block of size 9. Rule $2A'2$ is enabled on both robots that are at the extremity of the 1.block. When the scheduler activates such robots, a S^2-sequence is created. Rule $2A'6$ becomes enabled on the robots at distance 2 from the 2-towers. When such robots are activated by the scheduler, a configuration of type `intermediate` is created. Since we consider in this paper asynchronous robots, one tower can be created only at one extremity of the 1.block, Rule $2A'6$ becomes enabled on the robots at distance 2 from such a 2-tower. However, Rule $2A'2$ keeps being enabled on the other extremity (let the robot at the other extremity be r_1). Since we consider a fair scheduler, r_1 will be eventually enabled and a 2-tower is created. Rule $2A'6$ becomes enabled on the robots at distance 2 from the new 2-towers. Once such robot is activated, `intermediate` configuration is created.
- Other configurations. Two sub-cases are possible:
  - **Case (a)**: Robots at the extremity of the $\phi.group$ do not have any neighboring robots: Rule $2A'1$ is enabled on such robots. When the scheduler activates them, they become neighbor of an occupied node and we retrieve Case (b).
  - **Case (b)**: Robots at the extremity of the $\phi.group$ have one occupied node as a neighbor. Rule $2A'2$ becomes enabled on such robots. When the scheduler activates them, a 2-tower is created at at least one extremity of the $\phi.group$.

  When a 2-tower is created, four sub cases are possible as follow:
  - **Case (c)**: The tower does not have a neighboring occupied node (there is a sequence of nodes with multiplicities equal to 00201). In this case, Rule $2A'3$ becomes enabled on the robots part of the 2-tower. If the scheduler activates both robots at the same time then the 2-tower becomes neighbor to an occupied node. (i) If the scheduler activates only one robot, then the tower is destroyed and Rule $2A'2$ becomes enabled on the robot that did not move. When the rule is executed, the tower is built again and the configuration reached is exactly the same as in (i).
  - **Case (d)**: The Tower is neighbor of an occupied node that is part of a 1.block of size 3 (002111). Rule $2A'6$ is enabled in this case on the robot that is in the middle of the 1.block. Once the Rule is executed, another 2-tower is created.
  - **Case (e)**: The Tower is neighbor of an occupied node that is part of a 1.block of size 2 (002110). Rule $2A'5$ is enabled in this case on the robot that is in the extremity of the 1.block not having the 2-tower as a neighbor. When the rule is executed the robot that has moved becomes part of another 1.block and we retrieve Case (f).
  - **Case (f)**: The Tower is neighbor of an occupied node that has an empty node as a neighbor (00210). Rule $2A'4$ becomes enabled on the robot that is at distance 1 from the tower. Once it moves, it becomes neighbor of another occupied node and we retrieve either Case (c).

From the cases above we can deduce that a configuration with a sequence of node 2111 is reached in a finite time (Cases (c), (e) and (f)). In this case, Rule $2A'6$ becomes enabled on the robot that is in the middle of the 1.block of size 3 having a neighboring 2-tower (Case (d)). Once the robot moves, a 2-tower is created (2201). Note that the single robot in the sequence is not allowed to move unless it sees a 2-towers at distance 2 at each side. Thus, in the case the scheduler activates robots in only one extremity of the $\phi.group$, we are sure that robots in the other extremity will be eventually activated since they are the only one that will be allowed to move (the scheduler is fair). Hence, a symmetric configuration in which there will be one robot on the axes of symmetry and two 2-tower at each side of it is reached in a finite time. `intermediate` configuration is then reached and the lemma holds.

□

Lemma 4.2 *Starting from* `intermediate` *configuration,* `Middle` *configuration is eventually reached.*

Proof. When the configuration is of type `intermediate`, only $2A'7$ is enabled. Once the rule is executed, the robot that is not part of any tower becomes neigbor of a tower of size 2. `Middle` configuration is reached and the lemma holds. □

Lemma 4.3 *Starting from any towerless configuration that contains a single $\phi.group$, a configuration of type* Middle *is eventually reached.*

Proof. Directly follows from Lemmas 4.2 and 4.3. □

Lemma 4.4 *Starting from a configuration of type* `Middle`*,* `Terminal` *configuration is eventually reached.*

Proof. Let $T'1$ and $T'2$ be the two neigboring 2-towers that cannot see any other robot, and let $T1$ and $T2$ be the other 2-towers such as $T2$ has an neighboring occupied node other then $T1$ (let refer to the robot that is neighbor to $T1$ and not part of a 2-tower by $R1$) (refer to Figure 4.9). The sequence of robots containing $T1$, $T2$ and $R1$ is called *Explorer-Sequence*. On such a configuration, Rule $2A'8$ is enabled on $T1$. When the rule is executed, either (i) $T1$ becomes at distance 2 from $T2$ (the scheduler activates both robots part of $T1$ at the same time) or (ii) $T1$ is destroyed and in this case Rule $2A'9$ becomes the only rule enabled. When $2A'9$ is executed, $T1$ is restored and becomes at distance 2 from $T2$ (we retrieve Case (i)). On the new configuration Rule $2A'10$ is enabled on $T2$. When the rule is executed, either (a) $T2$ becomes neighbor to $T1$ (the scheduler activates both robots in $T1$) or (b) $T2$ is destroyed and Rule $2A'6$ becomes the only one enabled (the scheduler activates only one robot in $T2$). When the rule is executed, $T2$ is built again and becomes neighbor of $T1$ (we retrieve Case (a)). Rule $2A'11$ becomes then enabled on $R1$. When $R1$ is activated, *Explorer-Sequence* is built again, however, the distance between $T1$ and $T'1$ has decreased. Since the view of robots in the *Explorer-Sequence* is the same as in the `Middle` configuration, they will have the same behavior. Thus, at each time the distance between $T1$ and $T'1$ decreases. Hence $T1$ becomes eventually at distance 2 from $T'1$ in a finite time. Rule $2A'12$ becomes enabled on $T1$. If the scheduler activates only one robot in $T1$ then $T1$ is destroyed and Rule $2A'13$ becomes enabled, when the rule is executed, $T1$ becomes neighbor of $T'1$ (it is like the scheduler activates both robots at the same time). Since robots in $T2$ cannot see the 2-tower $T'1$, they will move towards $T1$, the same

holds for $R1$. Rule $2A'14$ is then enabled on $T2$, once it is executed, either a 4-tower is created and in this case the configuration is of type `Terminal` or 3-tower is created and in this case Rule $2A'15$ becomes enabled, when it is executed a 4-tower is created and the configuration becomes *Terminal* and the lemma holds. □

Lemma 4.5 *Starting from any towerless configuration that contains a single ϕ.group,* `Terminal` *configuration is eventually reached such that all nodes of the ring have been explored.*

Proof. From both lemma 4.3 and 4.4 we can deduce that starting from any initial configuration, `Terminal` is reached in a finite time. In another hand, from Lemma 4.3 we are sure that `Middle` configuration is reached in a finite time. Note that the only way to build the second towers ($T2$ and $T2'$) is to have a tower neighbor of a consecutive sequence of three robot. Since the distance between $R1$ and $T'2$ is equal to 3, we are sure that these nodes have been occupied and thus they have been explored. In another hand $T1$, $T2$ and $R1$ keep moving towards $T1'$ until they will become neighbor. Hence, all the nodes that were between $T1$ and $T1'$ have been all visited. We can deduce then that all the nodes of the ring have been explored and the lemma holds. □

4.2 Asynchronous Exploration using 7 robots

Recall that the initial configuration in this case contains a 1.block of size 7. The idea is similar to the one used in Algorithm 3. Let us first define two spacial configurations:

Definition 4.3 *A configuration is called* `Inter` *if there exists a sequence of seven nodes $u_i, u_{i+1}, \ldots, u_{i+5}, u_{i+6}$ (refer to Figure 4.11) such that:*

- $M_j = 2$ *for* $j \in \{i, i+1, i+6\}$
- $M_j = 0$ *for* $j \in \{i+3, i+4, i+5\}$
- $M_j = 1$ *for* $j = i+2$

Definition 4.4 *A configuration is called* `Final` *if there exists a sequence of four nodes $u_i, u_{i+1}, \ldots, u_{i+3}$ (refer to Figure 4.12) such that:*

- $M_j = 2$ *for* $j = i$
- $M_j = 4$ *for* $j = i+2$
- $M_j = 0$ *for* $j = i+1$
- $M_j = 1$ *for* $j = i+3$

The algorithm comprises two phases as follow:

1. `Preparation Phase:` The aim of this phase is to build an `Inter` Configuration. The starting configuration contains a single 1.block of size 7.
2. `Exploration Phase`: In this phase a set of robots are elected to perform the exploration task. At the end of the exploration `Final` Configuration is built to indicate the end of the exploration task.

Figure 4.11: $INTER$

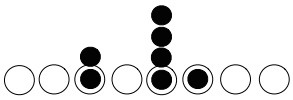


Figure 4.12: *Final* configuration

The formal description of the algorithm is given in Algorithm 4.

Algorithm 4 Asynchronous Exploration using 7 robots ($\phi = 2$)

`Preparation` Phase				
2A1:	01(1)11	::	$\leftarrow$	// *Move to my adjacent node having a free node as a neighbor*
2A2:	20(1)1?	::	$\leftarrow$	// *Move to my adjacent free node*
2A3:	10(1)01	::	$\leftarrow \vee \rightarrow$	// *Move to one of my neighboring node*
2A4:	00(1)20	::	$\leftarrow$	// *Move to my adjacent free node*
2A5:	10(1)02	::	$\leftarrow$	// *Move in the opposite direction of the tower*
2A6:	21(1)10	::	$\leftarrow$	// *Move toward the* 2.*tower*
2A7:	20(1)00	::	$\leftarrow$	// *Move towards the* 2.*tower*
`Exploration` Phase				
2A6:	21(1)10	::	$\leftarrow$	// *Move toward the* 2.*tower*
2A7:	20(1)00	::	$\leftarrow$	// *Move toward the* 2.*tower*
2A8:	00(2)21	::	$\leftarrow$	// *Move to my adjacent free node*
2A9:	01(1)21	::	$\leftarrow$	// *Move to my adjacent node having one free node as a neighbor*
2A10:	20(2)10	::	$\leftarrow$	// *Move to my adjacent free node*
2A11:	20(2)21	::	$\leftarrow$	// *Move to my adjacent free node*
2A12:	21(1)21	::	$\leftarrow$	// *Move in the opposite direction of the neighboring* 2.*tower*
2A13:	02(2)21	::	$\rightarrow$	// *Move to my adjacent node having one robot as a neighbor*
2A14:	02(1)32	::	$\rightarrow$	// *Move toward the* 3.*tower*

Proof of correctness We prove in the following the correctness of our algorithm.

Lemma 4.6 *Starting from a configuration that contains a single 1.block, a configuration of type* Inter *is eventually reached.*

Proof. When the configuration contains a single 1.block of size 7, Rule $2A1$ is enabled on the robots that are neighbor of the extremity of the 1.block. Once such robots move, a 2-tower is created at each extremity of the 1.block. Rule $2A2$ becomes enabled on the robots that are at distance 2 from the 2-towers. Once they move, they become neighbor of a 2-tower. Note that in the case the scheduler activates only robots at one extremity, we are sure that the configuration that will be reached in this case is exactly the same as when the scheduler activates both robots at each extremity since no robot will be able to move besides the ones at the other extremity. In another hand, since robots that did not move in the other extremity are not aware of the changes in the current configuration (recall that they are only able to see at distance 2), they will have the same behavior as when they are activated at the same time with the robots at the other extremity. Rule $2A3$ becomes then enabled. Since the view of the robot on which Rule $2A3$ is enabled, is symmetric the scheduler is the one that chooses the direction to take. Once such robot moves, it becomes neighbor of a single robot and it will be able to see one 2-tower. In the resulting configuration, Rule $2A4$ becomes enabled on the robot that is neighbor of a 2-tower having an empty node as a neighbor. Once this robot moves, it becomes able to see the other part that it could not see before moving. This robot continue to move towards the single robot until they become neighbor (refer to Rule $2A5$). Observe that one tower remains unseen by the other robots. The robot that is in the middle of the 1.block of size 3 that is neighbor of the tower of a 2-tower is the only one that can

move (refer to Rule 2A6). When this robot moves, a new 2.tower is created. The robot that is at distance 2 from the new 2.tower (note that there is only one such robot) is the only robot allowed to move, its destination is its adjacent empty node towards the 2.tower (refer to Rule 2A7). When it moves it becomes neighbor of the tower of size 2. `Inter` configuration is then reached and the Lemma holds. □

Lemma 4.7 *Starting from a configuration of type* `INTER`*, a configuration of type* `Final` *is eventually reached.*

Proof. When `Inter` configuration is reached, a subset of robots are elected to perform the exploration task. These robots are $T1$, $T2$ and $R1$ (refer to Figure 4.11). Observe that these robots form exactly the same sequence (*Explorer-Sequence*) as in `Middle` configuration (refer to Algorithm 3). It has been shown in Lemma 4.4 that this subset of robots keep moving towards $T'1$ until they become neighbor of $T'1$. Rule 2A13 becomes then enabled on the robots part of $T1$. If the scheduler activates both robots at the same time, a 4-tower is created and the configuration reached is `Final`. In the case the scheduler activates only one robot from $T1$ then a 3-tower is created. There will be one robot not part of a tower that is neighbor to the 3-tower. This robot is the one allowed to move (refer to Rule 2A14). Once the rule is executed a 4-tower is created and the configuration is *Final*. Thus, the lemma holds. □

Lemma 4.8 *Starting from a configuration that contains a single 1.block of size* 7*, a configuration of type Final is eventually reached such that all the nodes of the ring have been explored.*

Proof. From Lemma 4.6 and 4.7 we can deduce that starting from a configuration that contains a single 1.block, `Final` configuration is reached in a finite time. In another hand, when `Inter` configuration is reached, we are sure that the nodes between $R1$ and $T'1$ have been explored since these nodes have been occupied by robots at the beginning by the 1.block (refer to Figure 4.11). The sub set of robots that have performed the exploration task ($T1$, $T2$ and $R1$) keep moving towards $T'1$ until it becomes neighbor of $T'1$. Thus we are sure that the nodes that were between $T1$ and $T'1$ in `Inter` configuration have been visited. Thus all the nodes of the ring have been explored and the lemma holds. □

5 Visibility $\phi = 3$

In this section we first prove that no exploration is possible with 4 robots when $n > 13$. We then present two deterministic algorithms that solves the exploration. The first one uses 7 robots and works starting from any towerless configuration that contains a single ϕ.group. The second one uses 5 robots and works only when the starting configuration contains a single 1.block of size 5. In both solutions, we suppose that $n \geq k\phi + 1$.

5.1 Negative Results

In the following, let us consider a configuration γ_t in which there is a 3.group of size 4 (let call this configuration *Locked*). Let $u_i, u_{i+1}, \ldots, u_{i+9}$ be the nodes part of the 3.block in γ_t such as $u_i, u_{i+3}, u_{i+6}, u_{i+9}$ are occupied. Let r_1 and r_4 be the robots that are at the extremity of the 3.block such tha r_1 is on u_i and let r_2 and r_3 be the robots that are inside the 3.block such that r_2 is on u_{i+3}. The following three rules are the only ones can be enabled:

1. 000(1)001 :: $\leftarrow$
2. 000(1)001 :: $\rightarrow$
3. 100(1)001 :: $\leftarrow \vee \rightarrow$

Lemma 5.1 *Let $\mathcal{P}$ be a semi-synchronous exploration protocol for $\phi = 3$, $n > 13$, and $k < n$. Then, $\mathcal{P}$ does not include Rule 1.*

Proof. By contradiction, assume that $\mathcal{P}$ includes Rule 1. Note that Rule 1 is only enabled on r_1 and r_4. Assume that the adversary activates only r_1 inγ_t. Let us refer to the resulting configuration by ST (standing for "Second Trap" configuration). Once r_1 moves, it becomes an isolated robot on u_{i-1} in ST. u_{i+3} becomes the border of the 3.group. Assume that in ST, the adversary activates r_2 (on u_{i+1}) that executes Rule 1. Then, Configuration $ST + 1$ includes two 3.groups disjoint by three nodes. The former includes r_1 and r_2 (on u_{i-1} and u_i, respectively). The latter forms the sequence $r_3 \dots r_4$, located on $u_{i+6} \dots u_{i+9}$. Suppose that the scheduler activates r_3 that is on node u_{i+6} that executes Rule 1. Once it moves, the resulting configuration $ST + 2$ is undistinguishable from ST. A contradiction (Theorem 2.1). □

Lemma 5.2 *Let $\mathcal{P}$ be a semi-synchronous exploration protocol for $\phi = 3$, $n > 13$, and $k < n$. Then, $\mathcal{P}$ does not include Rule 2.*

Proof. By contradiction, assume that $\mathcal{P}$ includes Rule 2. Note that Rule 2 is only enabled on r_1 and r_4. Suppose that the scheduler activates both r_1 and r_4 at the same time. Let ST' be the reached configuration once such robots move. The following rules are then possible:

a. 010(1)001 :: $\leftarrow$

b. 010(1)001 :: $\rightarrow$

c. 000(1)010 :: $\rightarrow$

- **Case (a)**: Rule(a) is enabled. Assume that the scheduler activates both r_2 and r_3 at the same time. Once they move, all robots on the ring have the same view. (i) If they execute 000(1)100 :: $\rightarrow$ then suppose that the scheduler activates them all at the same time, then the configuration reached is undistinguishable from the previous one that contradicts Theorem 2.1. (ii) if they execute 000(1)100 :: $\leftarrow$. Assume that the scheduler activates only r_2 and r_3. Once the robots move the configuration reached is indistinguishable from ST'. A contradiction (Theorem 2.1).
- **Case (b)**: Rule (b) is enabled. Assume that the scheduler activates both r_2 and r_3 at the same time. r_2 and r_3 become then neighbors. The following Rules are possible:

  b1. 000(1)001 :: $\leftarrow$
  b2. 000(1)001 :: $\rightarrow$
  b3. 100(1)100 :: $\rightarrow$

  - **Case(b1)**. Rule (b1) is enabled. Suppose that the scheduler activates both r_1 and r_4 at the same time. Once the robots move they cannot see any occupied node (recall that $n > 13$). Thus if they execute rule 000(1)000 :: $\leftarrow \vee \rightarrow$, the robot move back to the previous location. A contradiction (Theorem 2.1).

- **Case(b2)**. Rule (b2) is enabled. Suppose that the scheduler activates both r_1 and r_4 at the same time. (i) If r_1 and r_4 execute rule 000(1)011 :: $\rightarrow$, a 1.block is created. If the robots inside the 1.block are the ones allowed to move, they either exchange their position (A contradiction, Theorem 2.1). Or two 2-towers are created. Suppose that the scheduler always activate robots part of the 2-tower at the same time. Thus, they will have the same behavior and act as a single robots. A contradiction (From [3]: no exploration is possible with only 2 robots even if the view is infinite). (ii) If rule 101(1)010 :: $\rightarrow$ enabled then, if the scheduler activates both robots at the same time no exploration is possible since whatever the robots that move an indistinguishable configuration is reached. A contradiction (Theorem 2.1).
- **Case(b3)**. Rule (b3) is enabled. Suppose that the scheduler activates r_2 and r_3 at the same time. They will simply exchange their position. The configuration that is reached is undistinguishable from the previous one. A contradiction (Theorem 2.1).

- **Case (c)**: Rule (c) is enabled. Once r_1 and r_4 move, two 1.blocks are created. If r_2 and r_3 are the ones allowed to move then if they execute 001(1)001:: $\leftarrow$, two towers are created and no exploration is possible (refer to [3]). If they execute 001(1)001:: $\rightarrow$, a new 1.block of size 2 is created. Note that in the reached configuration r_2 and r_3 cannot move anymore. If r_1 and r_4 execute 000(1)011 :: $\rightarrow$, a 1.block of size 4 is created and no exploration is possible (refer to Case (b)). If they execute 000(1)011 :: $\leftarrow$, they become at distance 3 from the 1.block. Let refer to the reached configuration by T'. If r_1 and $r4$ keep being enabled on T', they can only execute 000(1)001 :: $\leftarrow$. Suppose that the scheduler activates both robots at the same time. Once they move they cannot see any other robot in $T'+1$. (i) If 000(1)000 :: $\leftarrow \vee \rightarrow$ is enabled. Then suppose that the scheduler activates them at the same time such that they move back to their previous position. Thus, $T'+2$ is indistinguishable from $T'+1$. Contradiction (Theorem 2.1). (ii) if r_2 and r_3 are the enabled robots then if they execute 100(1)100 :: $\leftarrow$, then the configuration reached is indistinguishable from the configuration reached when Rule (b) is executed. Thus no exploration is possible in this case too.

□

Lemma 5.3 *Let $\mathcal{P}$ be a semi-synchronous exploration protocol for $\phi = 3$, $n > 13$, and $k < n$. Then, $\mathcal{P}$ does not include Rule 3.*

Proof.

By contradiction, suppose that the scheduler activates both r_2 and r_3 at the same time. Once they execute Rule 3, a 1.block is created. Such that robots in the 1.block cannot see any other robots. If r_2 and r_3 keep being enabled they can only move towards each other (otherwise the previous configuration is restored). Suppose that the scheduler activates them at the same time, the configuration reached is indistinguishable with the previous one. Contradiction (Theorem 2.1). If $r1$ and r_4 are the ones allowed to move then they can only execute 000(1)000 :: $\leftarrow \vee \rightarrow$ (recall that $n > 13$). Suppose that the scheduler activates them both at the same time such that they move in the opposite direction of the 1.block (let refer to the reached configuration by *IMP*). Since 000(1)000 :: $\leftarrow \vee \rightarrow$ keeps being enabled on both $r1$ and r_4. Suppose that the scheduler activates them at the same time such that they move back to their previous position. The configuration reached is indistinguishable from *IMP*. Contradiction (Theorem 2.1). □

Lemma 5.4 *No deterministic exploration is possible in the ATOM model for $\phi = 3$, $n > 13$ and $k < n$.*

Proof.
Follows from Lemmas 5.1, 5.2 and 5.3.

□

Lemma 5.5 *No deterministic exploration is possible in the CORDA model for $\phi = 3$, $n > 13$ and $k < n$.*

Proof. Follows directly from Lemma 5.4. □

5.2 Two Asynchronous Algorithms

In the following, we present two deterministic algorithms that solve the exploration problem. The first one uses 7 robots and works for any towerless initial configuration that contains a single ϕ.group such that $n \geq k\phi + 1$. The second one uses 5 robots but works only when the starting configuration contains a 1.block of size 5.

5.2.1 Exploration using $k = 7$.

Before detailing our solution, let us first define some special configurations:

Definition 5.1 *A configuration is called* `Set` *at instant t (refer to Figure 5.13) if there exists a sequence of 5 nodes $u_i, u_{i+1}, \ldots, u_{i+4}$ such that:*

- $M_j = 2$ *for* $j \in \{i+1, i+4\}$
- $M_j = 3$ *for* $j = i+1$
- $M_j = 0$ *for* $j \in \{i+2, i+3\}$

Definition 5.2 *A configuration is called* `Final` *at instant t (refer to Figure 5.14) if there exists a sequence of 4 nodes $u_i, u_{i+1}, \ldots, u_{i+3}$ such that:*

- $M_j = 2$ *for* $j \in \{i, i+1\}$
- $M_j = 0$ *for* $j = i+2$
- $M_j = 3$ *for* $j = i+3$

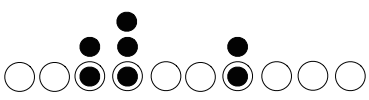


Figure 5.13: *Set* configuration

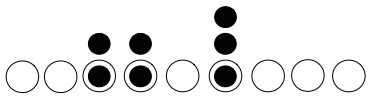


Figure 5.14: *Final* configuration

The algorithm comprises two phases as follow:

1. `Set-Up Phase`. The aim of this phase is to create a `Set Configuration`. The starting configuration contains a ϕ.group, such that $n \geq k(\phi) + 1$.

2. `Exploration Phase`. The starting configuration of this phase is the `Set Configuration`. A set of robots are elected to visit the ring's nodes. `Final Configuration` is created at the end of this phase.

The formal Description of the algorithm is given in Algorithm 5.

Algorithm 5 Asynchronous Exploration using 7 robots ($\phi = 3$)

`Set-Up` Phase				
3A1:	000(1)001	::	$\rightarrow$	*// Move toward the robot at distance* 3
3A2:	000(1)01?	::	$\rightarrow$	*// Move toward the robot at distance* 2
3A3:	000(1)1??	::	$\rightarrow$	*// Move toward my neighboring occupied node*
3A4:	000(2)001	::	$\rightarrow$	*// Move toward the robot at distance* 3
3A5:	000(2)01?	::	$\rightarrow$	*// Move toward the robot at distance* 2
3A6:	002(1)001	::	$\rightarrow$	*// Move in the opposite direction of the tower*
3A7:	002(1)01?	::	$\rightarrow$	*// Move in the opposite direction of the tower*
3A8:	021(1)001	::	$\rightarrow$	*// Move toward the robot at distance* 3
3A9:	021(1)01?	::	$\rightarrow$	*// Move toward the robot at distance* 2
3A10:	021(1)120	::	$\leftarrow \vee \rightarrow$	*// Move toward one of my neighboring node*
3A11:	220(1)200	::	$\leftarrow$	*// Move toward the tower at distance* 2
3A12:	022(1)020	::	$\leftarrow$	*// Move toward the neighboring tower*
`Exploration` Phase				
3A13:	000(2)300	::	$\leftarrow$	*// Move in the opposite direction of the* 3*.tower*
3A14:	001(1)300	::	$\leftarrow$	*// Move in the opposite direction of the* 3*.tower*
3A15:	000(2)030	::	$\leftarrow$	*// Move in the opposite direction of the* 3*.tower*
3A16:	001(1)030	::	$\leftarrow$	*// Move in the opposite direction of the* 3*.tower*
3A17:	200(3)000	::	$\leftarrow$	*// Move toward the* 2*.tower at distance* 3
3A18:	201(2)00?	::	$\leftarrow$	*// Move to my neighboring occupied node*
3A19:	202(1)00?	::	$\leftarrow$	*// Move to my neighboring occupied node*
3A20:	200(2)030	::	$\leftarrow$	*// Move in the opposite direction of the* 3*.tower*
3A21:	201(1)030	::	$\leftarrow$	*// Move towards the* 2*.tower at distance* 2
3A22:	000(2)020	::	$\rightarrow$	*// Move towards the* 2*.tower at distance* 2
3A23:	000(1)120	::	$\rightarrow$	*// Move towards my neighboring occupied node*

Proof of correctness. We prove in the following the correctness of Algorithm 5.

Lemma 5.6 *During the first phase, if there is a rule that is enabled on robot part of a* 2*.tower such that the scheduler activates only one robot in the* 2*.tower (the tower is destroyed), then the tower is built again and the configuration reached is exactly the same as when both robots in the* 2*.tower were activated at the same time by the scheduler.*

Proof. The rules that can be executed by the robots in a 2.tower in the first phase are Rules $3A4$ and $3A5$. In both Rules, robots in the tower can see only one robot at one side. Their destination is their adjacent empty node towards the robot that can be seen. Suppose that the scheduler activates only one robot in the 2.tower. In the configuration reached Rule $3A3$ is enabled. Note that the robot on which one of this rule is enabled is the one that was in the 2.tower and did not move. Once it execute $3A3$, it moves to its adjacent occupied node, thus a 2.tower is created again and the configuration reached is exactly the same as when both robots were activated by the scheduler. Hence the lemma holds. □

Lemma 5.7 *Starting from any towerless configuration that contains a single ϕ.group, a configuration containing an S^2-sequence is eventually reached.*

Proof. Two cases are possible as follow:

1. The starting configuration contains a single 1.block of size 7. Rule $3A3$ is then enabled on the two robots that are at the extremity of the 1.block. If the scheduler activates both robots that the same time, two 2.towers are created and the lemma holds. If the scheduler activates only one robot, then Rule $3A3$ is the only rule enabled in the system. When the rule is executed, the robot that was supposed to move moves to its neighboring occupied node and a 2.tower is created. Thus in this case too the lemma holds.

2. Other cases. Let consider only one extremity of the ϕ.group. If the robot at this extremity (let this robot be r_1) does not have a neighboring occupied node, then either Rule $3A1$ or $3A2$ is enabled. Once one of these rules is executed on r_1, r_1 becomes closer to an occupied node. Thus it becomes neighbor of an occupied node in a finite time. Rule $3A3$ becomes then enabled on r_1. Once r_1 moves, a 2.tower is created. If the 2.tower does not have any neighboring occupied node, either Rule $3A4$ or $3A5$ is enabled on robots in the 2.tower. According to Lemma 5.6, robots in the same 2.tower move eventually as when they are activated at the same time. By moving they become closer to an occupied node. Hence they become eventually neighbors of an occupied node. (i) If the robot that is neighbor of the 2.tower has an empty node as a neighbor then it moves to its adjacent empty node (refer to Rule $3A6$). (ii) in the case it has a neighboring occupied node besides the 2.tower, then if the robot on its neighboring occupied node move in the opposite direction of the 2.tower if it has a neighboring empty node (refer to Rules $3A7$ and $3A8$). By doing so the tower becomes neighbor of a 1.block of size 3. Robots at the other extremity have the same behavior since the only rules that can be executed are the ones that are enabled at the extremity. Thus, a configuration containing an S^2-sequence is reached in a finite time and the lemma holds.

□

Lemma 5.8 *Starting from a configuration containing an S^2-sequence,* `Set` *configuration is eventually reached.*

Proof. In a configuration containing an S^2-sequence, Rule $3A10$ is enabled. When the rule is executed, a new 2.tower is created. Rule $3A11$ becomes then enabled on the robot not part of a 2.tower (let this robot be r_1). Once it moves, only Rule $3A12$ becomes enabled on r_1. When the robot is activated a 3.tower is created. `Set` configuration is then reached and the lemma holds. □

Lemma 5.9 *Starting from a* `Set` *configuration,* `Final` *configuration is eventually reached.*

Proof. Let $T1$ and $T2$ be the 2.towers in the Set configuration such that $T1$ has a neighboring 3.tower that we call $T3$. Rule $3A13$ is enabled on the robots that are part of $T1$. If the scheduler activates only one robot, Rule $3A14$ becomes enabled on the robot that was supposed to move, thus the configuration reached is exactly the same as when both robots in $T1$ were activated at the same time. Rule $3A15$ becomes enabled on the $T1$. Once the robots on $T1$ are activated, the tower becomes at distance 3 from $T3$ (in the case the scheduler activates only one robot in $T1$, only Rule

$3A16$ becomes enabled on the robot that was supposed to move, once the scheduler activates the robot, the configuration reached is exactly the same as when both were activated at the same time). Robots on $T3$ are the only ones allowed to move (refer to Rule $3A17$). Once they move, $T3$ becomes at distance 2 from $T1$ (in the same manner, if the scheduler activates only some robots in $T3$ then either Rule $3A18$ or Rule $3A19$ is the only one enabled. Thus, $T3$ is built again (the configuration reached is similar to the one that was reached when all robots in the $T3$ have moved at the same time). Robots in $T1$ are now the only one allowed to move and so on. Observe that the distance between $T3$ and $T2$ increases while the distance between $T1$ and $T2$ decreases. When $T2$ and $T1$ becomes at distance 3. Rule $3A20$ is then enabled on $T1$ (if the scheduler activates only one robot then Rule $3A21$ is enabled and hence $T1$ is built again). Robots in $T2$ are now the only ones enabled (refer to Rule $3A22$), when they move, `Final` configuration is reached and the lemma holds (In the case the scheduler activates only one robot in $T2$, Rule $3A23$ becomes enabled on the robot that did not move. Once it is activated, `Final` configuration is reached and the lemma holds). □

Lemma 5.10 *Starting from a any towerless configuration that contains a single ϕ.group,* `Final` *configuration is eventually reached and all the nodes of the ring have been explored.*

Proof. From Lemmas 5.7, 5.8 and 5.9 we can deduce that `Final` configuration is eventually reached starting from a towerless configuration that contains a single ϕ.group. In another hand, the node between $T1$ and $T2$ in the `Set` configuration have been explored since they were occupied when the configuration contained an S^2-sequence. When the set of robots in charge of performing the exploration task move on the ring, the distance between $T1$ and $T2$ decreases such that when `Final` configuration is reached $T1$ and $T2$ become neighbor. Thus, we are sure that all the nodes of the ring has been explored and the lemma holds. □

5.2.2 Exploration using $k = 5$.

In the following, we provide two definitions that are needed in the description of our algorithm:

Definition 5.3 *A configuration is called* `Set2` *at instant t (refer to Figure 5.15) if there exists a sequence of 6 nodes $u_i, u_{i+1}, \ldots, u_{i+5}$ such that:*

- $M_j = 1$ *for* $j \in \{i, i+5\}$
- $M_j = 3$ *for* $j = i+1$
- $M_j = 0$ *for* $j \in \{i+2, i+3, i+4\}$

Definition 5.4 *A configuration is called* `Done` *at instant t (refer to Figure 5.16) if there exists a sequence of 4 nodes $u_i, u_{i+1}, \ldots, u_{i+3}$ such that:*

- $M_j = 1$ *for* $j \in \{i, i+1\}$
- $M_j = 0$ *for* $j = i+2$
- $M_j = 3$ *for* $j = i+3$

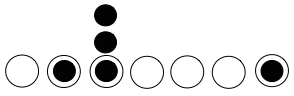


Figure 5.15: *Set*2

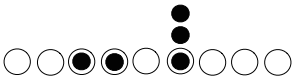


Figure 5.16: *Done* configuration

The algorithm comprises also two phases as follow:

1. `Set-Up Phase`. The initial configuration of this phase contains a ϕ.group such that $n \geq k\phi+1$. The aim of this phase is to create a `Set2` configuration.

2. `Exploration Phase`. The starting configuration of this phase is `Set2` Configuration. A set of robots are elected to perform the exploration task. At the end of this phase, `Done` configuration is created.

The formal description of our algorithm is given in Algorithm 6.

Algorithm 6 Asynchronous Exploration using 5 robots ($\phi = 3$)

`Set-Up` Phase				
3A'1:	011(1)110	::	$\rightarrow \vee \leftarrow$	// *Move to one of my neighboring node*
3A'2:	120(1)100	::	$\leftarrow$	// *Move to towards the tower at distance* 2
3A'3:	012(1)010	::	$\leftarrow$	// *Move to the tower*
3A'4:	300(1)000	::	$\rightarrow$	// *Move in the opposite direction of the* 3.*tower*
3A'5:	120(1)101	::	$\leftarrow$	// *Move toward the* 2.*tower (case* $n = 6$*)*
3A'6:	300(1)001	::	$\rightarrow$	// *Move toward in the opposite direction of the* 3.*tower (case* $n = 7$*)*
`Exploration` Phase				
3A'7:	000(1)300	::	$\leftarrow$	// *Move in the opposite direction of the* 3.*tower*
3A'8:	000(1)030	::	$\leftarrow$	// *Move in the opposite direction of the* 3.*tower*
3A'9:	100(3)000	::	$\leftarrow$	// *Move toward the isolated robot*
3A'10:	101(2)000	::	$\leftarrow$	// *Move to my neighboring occupied node*
3A'11:	102(1)000	::	$\leftarrow$	// *Move to the* 2.*tower*
3A'12:	100(1)030	::	$\leftarrow$	// *Move in the opposite direction of the* 3.*tower*
3A'13:	010(1)030	::	$\leftarrow$	// *Move in the opposite direction of the* 3.*tower*
3A'14:	010(1)300	::	$\leftarrow$	// *Move in the opposite direction of the* 3.*tower (case* $n = 6 \wedge n = 7$ *)*

Proof of Correctness We prove in the following the correctness of our solution.

Lemma 5.11 *Starting from a configuration that contains a* 1.*block of size* 5 *such that* $n > k + 2$, `Set2` *configuration is eventually reached.*

Proof. When the configuration contains a 1.block of size 5, Rule $3A'1$ is enabled. Once the rule is executed a 2.tower is created. On the robot that is at distance 2 from the 2.tower (let this robot be r_1) Rule $3A'2$ is enabled. When the rule is executed r_1 becomes neighbor of the 2.tower. Rule $3A'3$ becomes the only one enabled then enabled. When r_1 is activated, it executes the rule and it joins the 2.tower and hence a 3.tower is created. Rule $3A'4$ becomes the only rule enabled on the robot that is at distance 3 from the 3.block, when the robot is activated, it becomes at distance 4 from the 3.tower. The `Set2` configuration is then created and the lemma holds.

□

Lemma 5.12 *Starting from a configuration of type* `Set2`, `Done` *configuration is eventually reached.*

Proof.

Let r_1 be the robot that is neighbor of the 3.tower and let r_2 be the robot that cannot see any other robot. r_1 is the only robot allowed to move and this until it becomes at distance 3 from the tower (see Rules $3A'7$ and $3A'8$). Rule $3A'9$ becomes enabled on the robots part of the 3.tower, their destination is their adjacent empty node towards r_1. In the case the scheduler activates some robots in the tower, the remaining robots (the one that were supposed to move) are the only ones allowed to move, their destination is their adjacent occupied node (see Rule $3A'10$ and $3A'11$). Thus the configuration reached is exactly the same as when all the robots that are in the 3.tower are activated by the scheduler at the same time. Hence, the tower becomes eventually at distance 2 from r_1. r_1 is now allowed to move, its destination is its adjacent empty node in the opposite direction of the 3.tower. Note that once it moves it becomes at distance 3 from this tower. Thus, robots in the 3.tower are the ones allowed to move and so on. Both the 3.tower and r_1 keep moving in the same direction such that at each time they move they become closer to r_2 (the distance between r_1 and r_2 decreases). Thus, r_1 and r_2 becomes eventually neighbors. Note that the robots in the 3.tower cannot see r_2 yet, so it will continue to move towards r_1. When all of them move, *Done* configuration is reached and the lemma holds. □

Lemma 5.13 *Starting from a configuration that contains a 1.block, when* `Done` *configuration is reached, and all the nodes of the ring have been explored.*

Proof. From Lemmas 5.11 and 5.12 we deduce that staring from any towerless configuration that contains a single 1.block such that $n > k + 2$, `Done` configuration is reached in a finite time. Let T be the 3.tower and let r_1 (resp r_2) be respectively the robots that is neighbor of the 3.tower (resp the robot that cannot see any other robot). In `Set2` configuration, the nodes between the T and r_2 has been already visited since the starting configuration was a 1.block. T and r_1 keep moving in the same direction such that the distance between r_1 and r_2 decreases at each time. When `Done` configuration is reached, r_1 and r_2 become neighbor. We can then deduce that all the nodes of the ring have been explored and the lemma holds. □

6 Conclusion

In this paper, we studied the exploration of uniform rings by a team of oblivious robots. The assumptions of unlimited visibility made in previous works has enabled them to focus only on overcoming the computational weaknesses of robots introduced by the simultaneous presence of obliviousness and asynchrony in the design of exploration algorithms. In this paper, we added one more weakness: *Myopia ie,* robots have only a limited visibility. We provided evidences that the exploration problem can still be solved under some conditions by oblivious robots despite myopia. We studied the problem for both synchronous and asynchronous settings, and considered three types of visibility capabilities: $\phi = 1$, $\phi = 2$, and $\phi = 3$.

The complete characterization for which the exploration of the ring by myopic robots is solvable remains open in general. We conjuncture that the solutions proposed in this paper are optimal with respect to number of robots. We also believe that the condition $n > k\phi+1$ is a necessary condition to solve the problem. Also, the problem of exploring other topologies and arbitrary graphs by myopic robots is a natural extension of this work.

References

[1] H Ando, Y Oasa, I Suzuki, and M Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transaction on Robotics and Automation*, 15(5):818–828, 1999.

[2] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. On the computational power of oblivious robots: forming a series of geometric patterns. In *29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 267–276, 2010.

[3] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 195–208, 2009.

[4] Yoann Dieudonné, Ouiddad Labbani-Igbida, and Franck Petit. Circle formation of weak mobile robots. *ACM Transactions on Adaptive and Autonomous Systems (TAAS)*, 3(4), 2008.

[5] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *14th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 105–118, 2007.

[6] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci.*, 411(14-15):1583–1598, 2010.

[7] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337(1-3):147–168, 2005.

[8] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theor. Comput. Sci.*, 407(1-3):412–447, 2008.

[9] Samuel Guilbault and Andrzej Pelc. Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. In *18th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 162–173, 2011.

[10] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 101–113, 2010.

[11] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *18th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 150–161, 2011.

[12] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In *12th International Conference on Principles of Distributed Systems (OPODIS)*, pages 446–462, 2008.

[13] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science*, 390(1):27–39, 2008.

[14] Anissa Lamani, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal deterministic ring exploration with oblivious asynchronous robots. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 183–196, 2010.

[15] G. Prencipe. CORDA: Distributed coordination of a set of autonomous mobile robots. In *Proc. 4th European Research Seminar on Advances in Distributed Systems (ERSADS'01)*, pages 185–190, Bertinoro, Italy, May 2001.

[16] G Prencipe. *Distributed Coordination of a Set of Autonomous Mobile Robots.* PhD thesis, Dipartimento di Informatica, University of Pisa, 2002.

[17] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.